FIG. 1

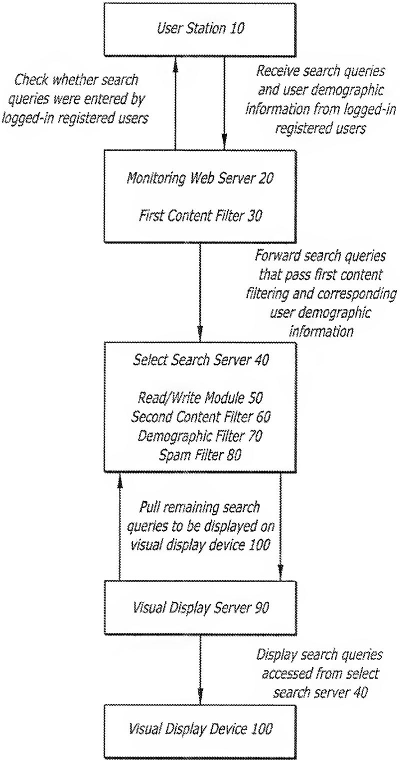
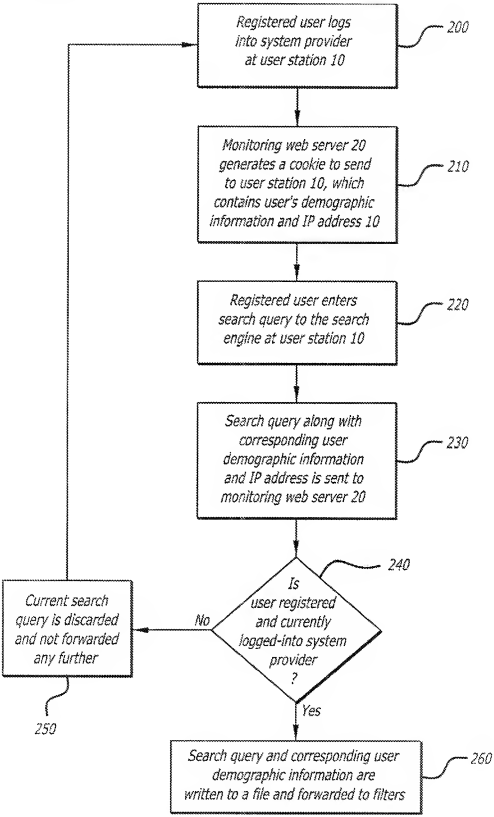

User Station 10
Check whether search queries were entered by logged-in registered users
Receive search queries and user demographic information from logged-in registered users
Monitoring Web Server 20
First Content Filter 30
Forward search queries that pass first content filtering and corresponding user demographic information
Select Search Server 40
Read/Write Module 50
Second Content Filter 60
Demographic Filter 70
Spam Filter 80
Pull remaining search queries to be displayed on visual display device 100
Visual Display Server 90
Display search queries accessed from select search server 40
Visual Display Device 100

FIG. 2


Registered user logs into system provider at user station 10
200
Monitoring web server 20 generates a cookie to send to user station 10, which contains user's demographic information and IP address 10
210
Registered user enters search query to the search engine at user station 10
220
Search query along with corresponding user demographic information and IP address is sent to monitoring web server 20
230
Is user registered and currently logged-into system provider ?
240
No
Current search query is discarded and not forwarded any further
250
Yes
Search query and corresponding user demographic information are written to a file and forwarded to filters
260

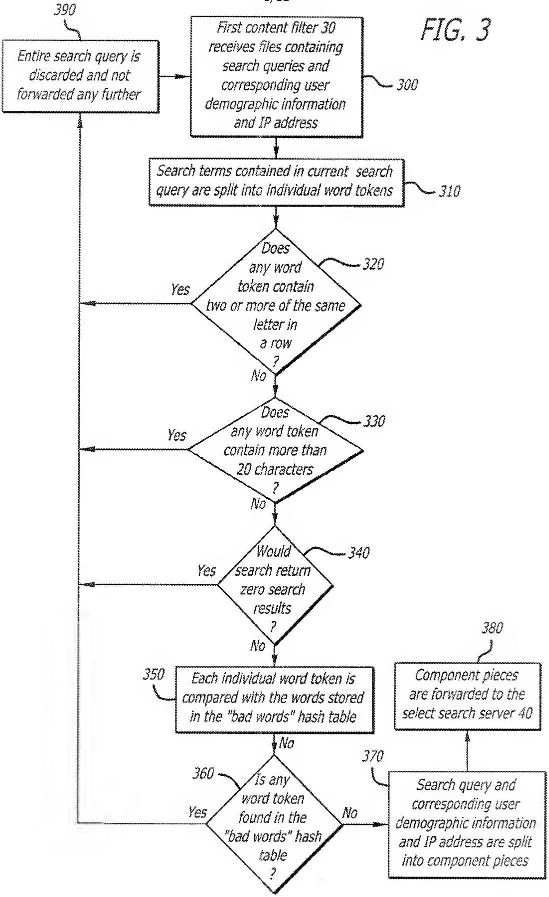

390
Entire search query is discarded and not forwarded any further
First content filter 30 receives files containing search queries and corresponding user demographic information and IP address
300
Search terms contained in current search query are split into individual word tokens
310
Does any word token contain two or more of the same letter in a row ?
320
Yes
No
Does any word token contain more than 20 characters ?
330
Yes
No
Would search return zero search results ?
340
Yes
No
350
Each individual word token is compared with the words stored in the "bad words" hash table
No
360
Is any word token found in the "bad words" hash table ?
Yes
No
370
Search query and corresponding user demographic information and IP address are split into component pieces
380
Component pieces are forwarded to the select search server 40

FIG. 3

FIG. 4

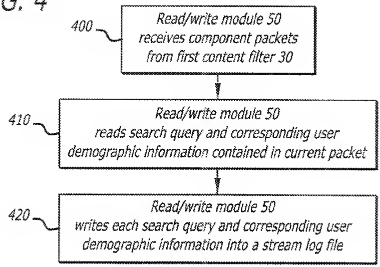

400
Read/write module 50 receives component packets from first content filter 30
410
Read/write module 50 reads search query and corresponding user demographic information contained in current packet
420
Read/write module 50 writes each search query and corresponding user demographic information into a stream log file

FIG. 5

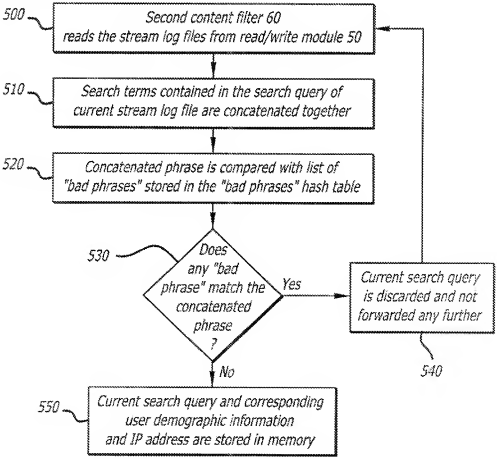

500
Second content filter 60 reads the stream log files from read/write module 50
510
Search terms contained in the search query of current stream log file are concatenated together
520
Concatenated phrase is compared with list of "bad phrases" stored in the "bad phrases" hash table
530
Does any "bad phrase" match the concatenated phrase ?
Yes
Current search query is discarded and not forwarded any further
540
No
550
Current search query and corresponding user demographic information and IP address are stored in memory

FIG. 6

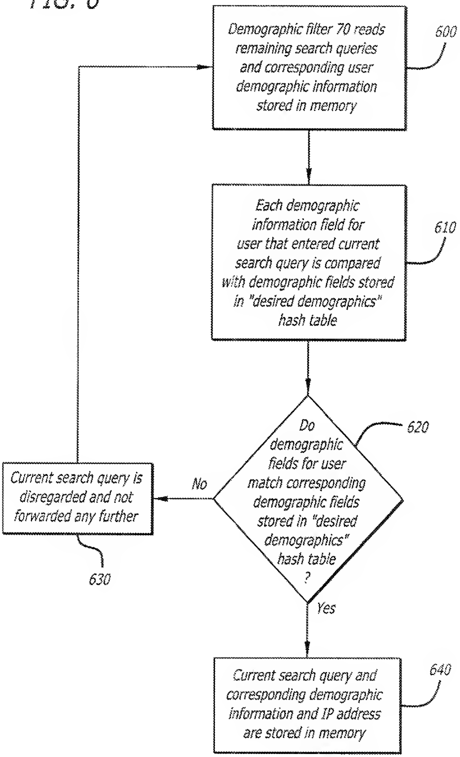

Demographic filter 70 reads remaining search queries and corresponding user demographic information stored in memory
600
Each demographic information field for user that entered current search query is compared with demographic fields stored in "desired demographics" hash table
610
Do demographic fields for user match corresponding demographic fields stored in "desired demographics" hash table ?
620
No
Current search query is disregarded and not forwarded any further
630
Yes
Current search query and corresponding demographic information and IP address are stored in memory
640

FIG. 7

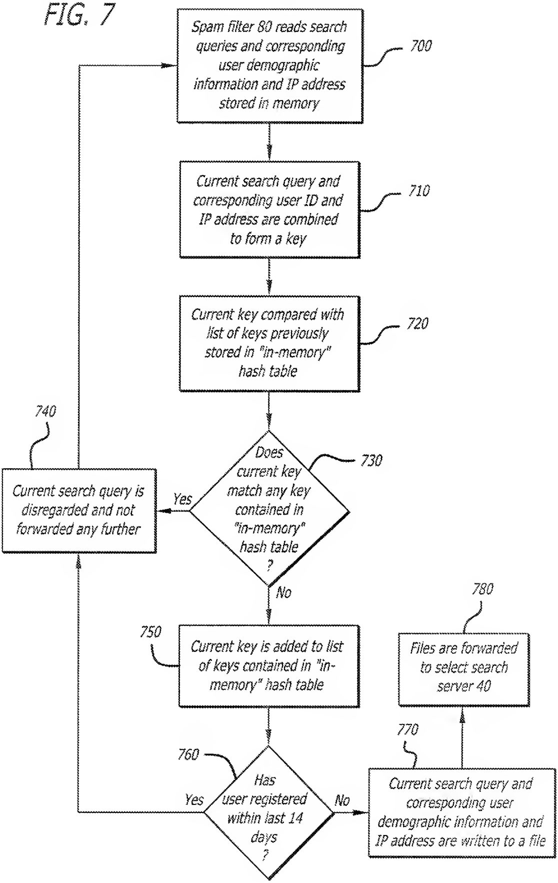

Spam filter 80 reads search queries and corresponding user demographic information and IP address stored in memory
700
Current search query and corresponding user ID and IP address are combined to form a key
710
Current key compared with list of keys previously stored in "in-memory" hash table
720
Does current key match any key contained in "in-memory" hash table ?
730
Yes
Current search query is disregarded and not forwarded any further
740
No
Current key is added to list of keys contained in "in-memory" hash table
750
780
Files are forwarded to select search server 40
770
760
Has user registered within last 14 days ?
Yes
No
Current search query and corresponding user demographic information and IP address are written to a file

FIG. 8

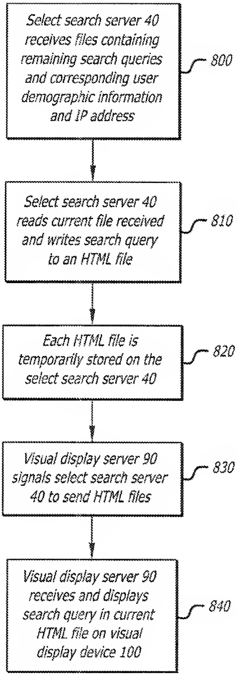

Select search server 40 receives files containing remaining search queries and corresponding user demographic information and IP address
800
Select search server 40 reads current file received and writes search query to an HTML file
810
Each HTML file is temporarily stored on the select search server 40
820
Visual display server 90 signals select search server 40 to send HTML files
830
Visual display server 90 receives and displays search query in current HTML file on visual display device 100
840

FIG. 9A

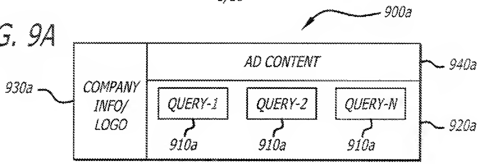


FIG. 9B

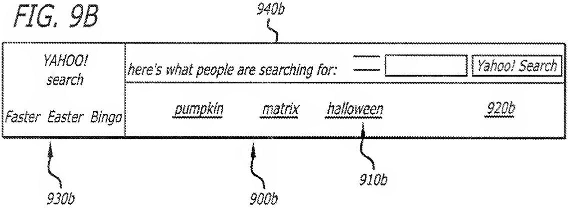
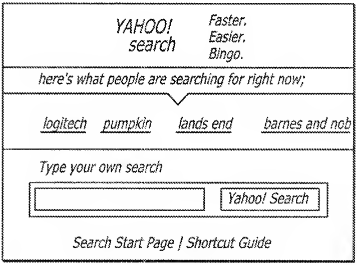


FIG. 9C



```
// shim will look for the following variables which will be supplied by the ad unit:
// nqIn  = number of terms to get from the CGI - required!
// ageIn  = age range (all:0-150/default, 1:0-18, 2:18-25, 3:25-35, 4:35-50, 5:50-150)
// genIn  = gender (m-male, f-female, a-all) - not required
// zipIn = zip code - not required
// radIn = radius around zip-code - not required
```

BLOCK 1:

```
var extraParams = "";
if ( (nqIn < 1) or (nqIn == undefined) ) nqIn = 20;
if ( ageIn != undefined ) { extraParams += "&age="+ageIn; }
if ( genIn != undefined ) { extraParams += "&gen="+genIn; }
if ( zipIn != undefined ) { extraParams += "&zip="+zipIn; }
if ( radIn != undefined ) { extraParams += "&rad="+radIn; }
```

BLOCK 2:

```
baseURL = "http://select.search.server.com/terms?nq="+nqIn+"&r=x";
extraParams += "&rnd="+Math.random();
baseURL += extraParams;
```

BLOCK 3:

```
buzz = new XML();
buzz.onLoad = parseResults;
buzz.ignoreWhite = true;
var listing=[];
buzz.load(baseURL);
```

BLOCK 4:

```
 var dataState = "loading";
```

BLOCK 5:

```
function parseResults(result) {
        if (result) {
                var items = buzz.firstChild;
                for (i=1; i<items.length; i++) {
                        listing[i-1] = new Object();
                        listing[i-1]["keyword"] = items[i].childNodes[0];    }
                dataState = "available";    }
        else
        {               dataState = "unavailable";        }}
```

FIG. 10

```
// This SWF looks for two variables
// delta = step size to take; dy=20 yields 2.0 pixels per
//frame
// offset = distance between keywords; can be negative to
// tighten up spacing between keyword blocks
// If they are not found, default values are set in frame
// 15 of this level.
// ttw = "time to wait" in seconds - defaults to 5 seconds
// (in this frame--see below)

// shim.swf will look for variables as follows:
// n  = number of terms to get from the CGI - defaults to 20 if not set
// a  = age range (all:0-150/default, 1:0-18, 2:18-25, 3:25-35, 4:35-50, 5:50-150)
// gen = gender (m-male, f-female, a-all) - not required
// zip = zip code - not required
// rad = radius around zip-code - not required
//
// EXAMPLE:
// To get a scroll of 20 keywords from users in ZIP code
//94089, load this scroller as follows:
//
ticker.loadMovie("http://path_to_scroller_SWF/vscroll_300x300.swf?n=20&zc=9408
9"
```

BLOCK 1:

```
pShim.loadMovie("http://select.search.server.com/shim.swf?nqIn="+ nq
+"&ageIn="+a+"&genIn="+gen+"&zipIn="+zip+"&radIn="+rad)
```

BLOCK 2:

```
var scrollStatus = "loading"
```

BLOCK 3a:

```
var startTime = getTimer()
```

BLOCK 3b:

```
if ( ttw == undefined ){ ttw = 5;}
```

FIG. 11

BLOCK 1:

```
if ( ttw*1000 < (startTime - getTimer()) )

{
   scrollStatus = "unavailable";
   goToAndStop(15);
}
else if ( pShim.dataState == "loading" )
```

BLOCK 2:

```
{
   gotoAndPlay(2);
   scrollStatus = "loading"
}
else
```

BLOCK 3:

```
{
   scrollStatus = pShim.dataState;
   gotoAndStop(15);
}
```

FIG. 12

```
// delta = step size to take; dy=20 yields 2.0 pixels per frame
// offset = distance between keywords; can be negative to
// tighten up spacing between keyword blocks
BLOCK 1:
if ( delta == undefined ){ delta = 20;}
BLOCK 2:
if ( offset == undefined ){ offset = 0;}
BLOCK 3:
offset = 1.0*offset;// coerce from string to number, just in case
BLOCK 4:
initMove=move=delta/10

BLOCK 5:
isMoving=true

BLOCK 6:
function hmove(mc){
   if(!isMoving) {
      move=0
   } else (move=initMove     )

BLOCK 6a:
mc._x -= move

BLOCK 6b:
if(mc._x < -mc._width){
      mc._x+=2*xPos;      }
   mc._x= Math.floor(mc._x)   }

BLOCK 7:
stop();

BLOCK 8a:
hoverColor="FF0000"
BLOCK 8b:
regularColor="0000FF"
```

FIG. 13a

BLOCK 9:

```
searchURL = "http://search.server.com/search?p=";
```

BLOCK 10:

```
if ( scrollStatus == "available" )
   {
    var localListing = [];
    localListing = pShim.listing;
    formatResults(localListing);
   }
```

BLOCK 11:

```
function formatResults(data) {
      xPos=0
      for (i=0; i<data.length; i++) {
         buzzMC1.attachMovie("item", "b"+i, i);
         buzzMC2.attachMovie("item", "b"+i, i);
         var mc1 = buzzMC1["b"+i];
         var mc2 = buzzMC2["b"+i];
         var head = data[i].keyword;
         var url = searchURL+escape(head);
         mc1.u = mc2.u=url;
         mc1.keyword = mc2.keyword = head;
         mc1.head = mc2.head="<font
color='#"+regularColor+"'><u>"+head+"</u></font>";
         mc1.txt = mc2.txt= head
         var txtWidth=pixelWidthArial(head, 10);
         mc1.buttonMC._width=mc2.buttonMC._width = txtWidth
         mc1._x = mc2._x=xPos;
         xPos += txtWidth+offset
      }
      buzzmc2._x +=xPos
   }
```

FIG. 13b

BLOCK 1:

```
on(rollOver){
   _parent._parent._parent.isMoving=false
   _parent.head = "<font
color='#"+_parent._parent._parent.hoverColor+"'>"+_parent.txt + "</font>"
  }
```

BLOCK 2:

```
on(rollOut, dragOut){ // Block 2
     _parent._parent._parent.isMoving=TRUE
     _parent.head = "<font
color='#"+_parent._parent._parent.regularColor+"'><u>"+_parent.txt + "</u></font>"
  }
```

BLOCK 3:

```
on(release){
   // function doClick(keyword) must be defined in the _root level or nothing happens
   _root.doClick(_parent.keyword)
  }
```

FIG. 14

FIG. 15

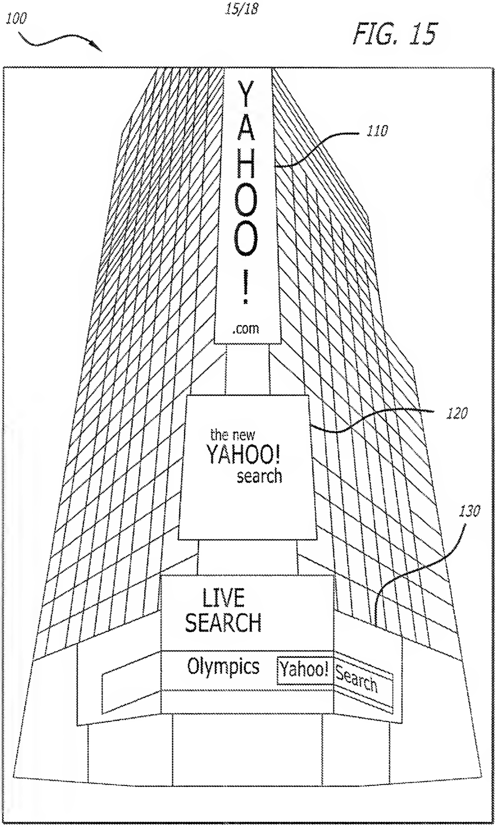

100
110
120
130
YAHOO!
.com
the new
YAHOO!
search
LIVE
SEARCH
Olympics
Yahoo!
Search

FIG. 16

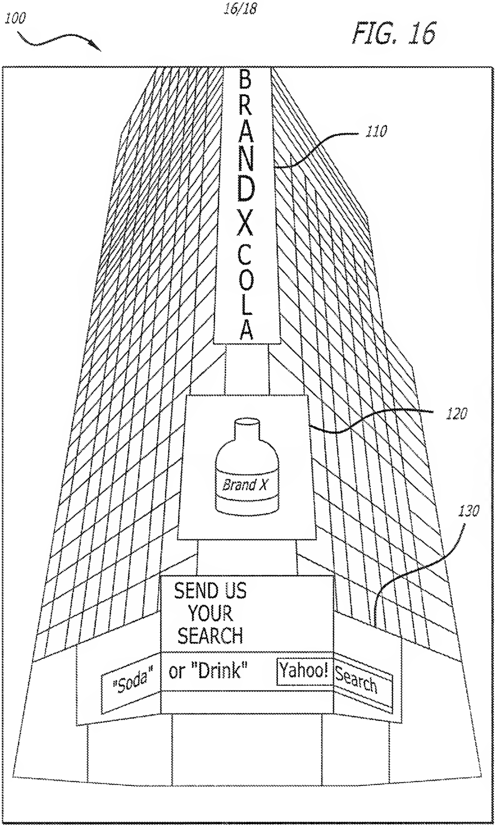
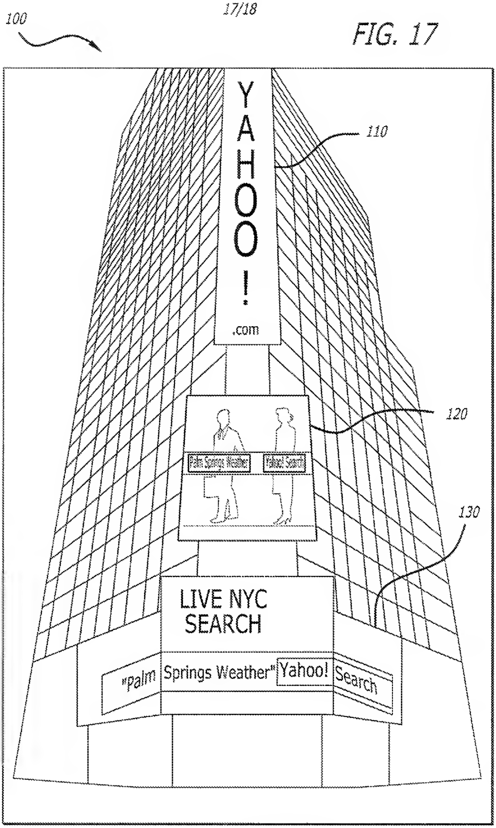

100
110
120
130
BRAND X COLA
Brand X
SEND US YOUR SEARCH
"Soda"
or "Drink"
Yahoo!
Search

FIG. 17


100
YAHOO!
.com
110
120
Palm Springs Weather
Yahoo! Search
130
LIVE NYC SEARCH
"Palm Springs Weather" Yahoo! Search

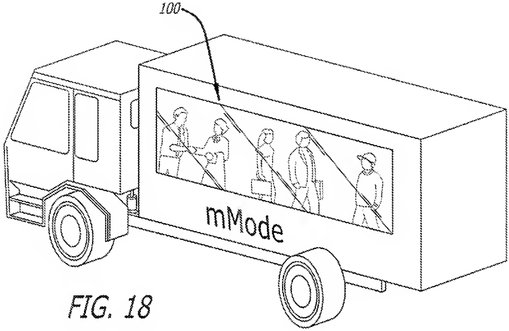


FIG. 18

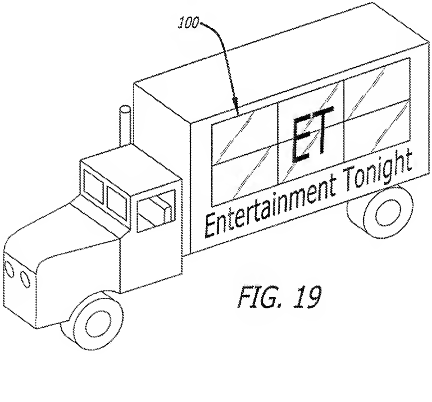


FIG. 19